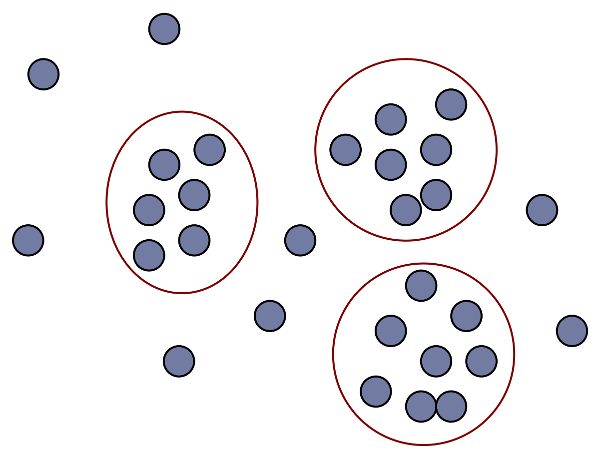
Chapter 3: Cluster Analysis

- **3.1 Basic Concepts of Clustering**
  - 3.1.1 Cluster Analysis
  - 3.1.2 Clustering Categories
- **3.2 Partitioning Methods**
  - 3.2.1 The principle
  - 3.2.2 K-Means Method
  - 3.2.3 K-Medoids Method
  - 3.2.4 CLARA
  - 3.2.5 CLARANS
- **3.3 Hierarchical Methods**
- **3.4 Density-based Methods**
- **3.5 Clustering High-Dimensional Data**
- **3.6 Outlier Analysis**

3.1.1 Cluster Analysis

- Unsupervised learning (i.e., Class label is unknown)
- Group data to form new categories (i.e., clusters), e.g., cluster houses to find distribution patterns
- Principle: Maximizing intra-class similarity & minimizing interclass similarity



- **Typical Applications**
  - → WWW, Social networks, Marketing, Biology, Library, etc.

3.1.2 Clustering Categories

- **Partitioning Methods**
  - → Construct k partitions of the data
- **Hierarchical Methods**
  - → Creates a hierarchical decomposition of the data
- **Density-based Methods**
  - → Grow a given cluster depending on its density (# data objects)
- **Grid-based Methods**
  - → Quantize the object space into a finite number of cells
- **Model-based methods**
  - → Hypothesize a model for each cluster and find the best fit of the data to the given model
- **Clustering high-dimensional data**
  - → Subspace clustering
- **Constraint-based methods**
  - → Used for user-specific applications

Chapter 3: Cluster Analysis

- **3.1 Basic Concepts of Clustering**
  - 3.1.1 Cluster Analysis
  - 3.1.2 Clustering Categories
- **3.2 Partitioning Methods**
  - 3.2.1 The principle
  - 3.2.2 K-Means Method
  - 3.2.3 K-Medoids Method
  - 3.2.4 CLARA
  - 3.2.5 CLARANS
- **3.3 Hierarchical Methods**
- **3.4 Density-based Methods**
- **3.5 Clustering High-Dimensional Data**
- **3.6 Outlier Analysis**

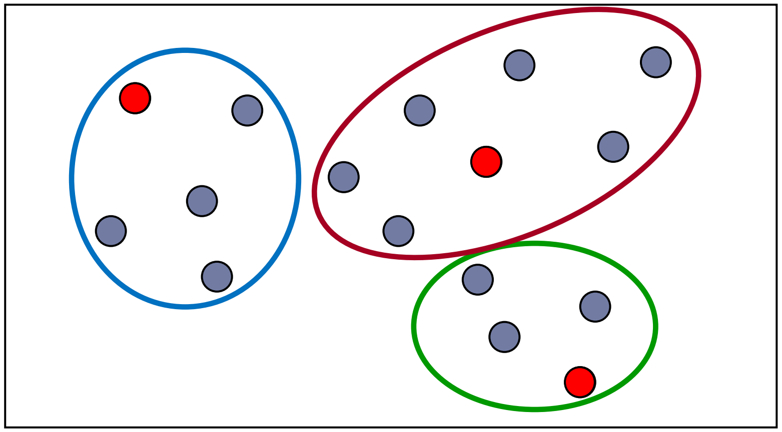
3.2.1 Partitioning Methods: The Principle

- Given
  - A data set of **n** objects
  - **K** the number of clusters to form
- Organize the objects into k partitions **(k<=n)** where each partition represents a cluster
- The clusters are formed to optimize an objective partitioning criterion
  - Objects within a cluster are **similar**
  - Objects of different clusters are **dissimilar**

3.2.2 K-Means Method

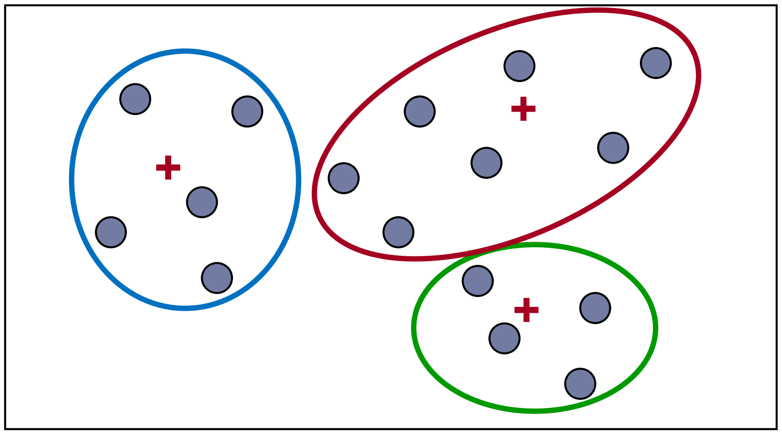
Choose 3 objects (cluster centroids)

Assign each object to the closest centroid to form Clusters



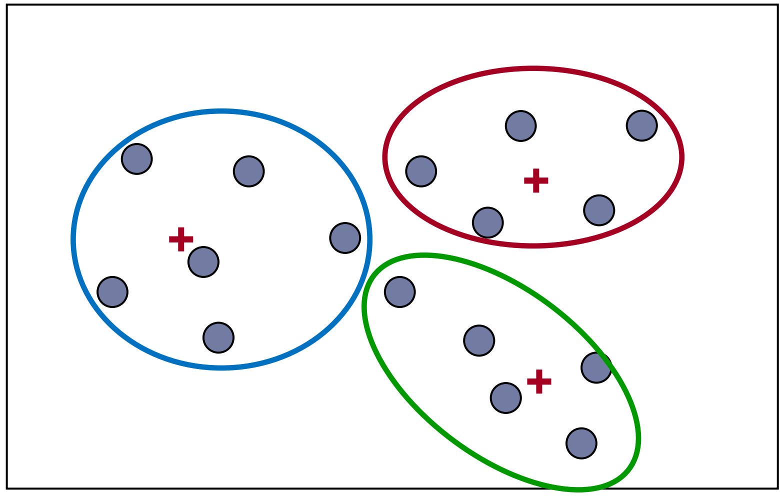
Goal: create 3 clusters (partitions)

Update cluster centroids

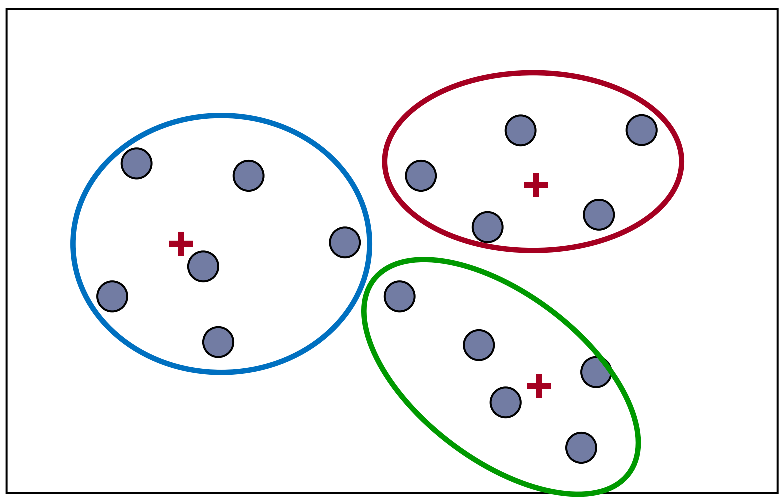


K-Means Method

Recompute Clusters



If Stable centroids, then stop



K-Means Algorithm

- **Input**
  - K: the number of clusters
  - D: a data set containing n objects
- **Output**: A set of k clusters
- **Method**:

(1) Arbitrary choose k objects from D as in initial cluster centers
(2) **Repeat**
(3) Reassign each object to the most similar cluster based on the mean value of the objects in the cluster
(4) Update the cluster means
(5) **Until** no change

K-Means Properties

- The algorithm attempts to determine k partitions that minimize the square-error function

$$E = \sum_{i-1}^{k} \sum_{p \in C_i} (p - m_i)^2$$

  - → **E**: the sum of the squared error for all objects in the data set
  - → **P**: the data point in the space representing an object
  - → **m_i**: is the mean of cluster C_i

- It works well when the clusters are compact clouds that are rather well separated from one another

K-Means Properties

Advantages

- K-means is relatively scalable and efficient in processing large data sets
- The computational complexity of the algorithm is **O(nkt)**
  - → **n:** the total number of objects
  - → **k:** the number of clusters
  - → **t:** the number of iterations
  - → **Normally:** k<<n and t<<n

Disadvantage

- Can be applied only when the mean of a cluster is defined
- Users need to specify k
- K-means is not suitable for discovering clusters with nonconvex shapes or clusters of very different size
- It is sensitive to noise and outlier data points (can influence the mean value)

Variations of the K-Means Method

- A few variants of the ***k-means*** which differ in
  - Selection of the initial *k* means
  - Dissimilarity calculations
  - Strategies to calculate cluster means
- Handling categorical data: ***k-modes*** (Huang'98)
  - Replacing means of clusters with modes
  - Using new dissimilarity measures to deal with categorical objects
  - Using a frequency-based method to update modes of clusters
  - A mixture of categorical and numerical data

3.2.3 K-Medoids Method

- Minimize the sensitivity of k-means to outliers
- Pick actual objects to represent clusters instead of mean values
- Each remaining object is clustered with the representative object (**Medoid**) to which is the most similar
- The algorithm minimizes the sum of the dissimilarities between each object and its corresponding reference point

$$E = \sum_{i-1}^{k} \sum_{p \in C_i} | p - o_i |$$

  - **E**: the sum of absolute error for all objects in the data set
  - **P**: the data point in the space representing an object
  - **O_i**: is the representative object of cluster C_i

K-Medoids Method: The Idea

- Initial representatives are chosen randomly

- The iterative process of replacing representative objects by no representative objects continues as long as the quality of the clustering is improved

- For each representative Object O

  → For each non-representative object R, swap O and R

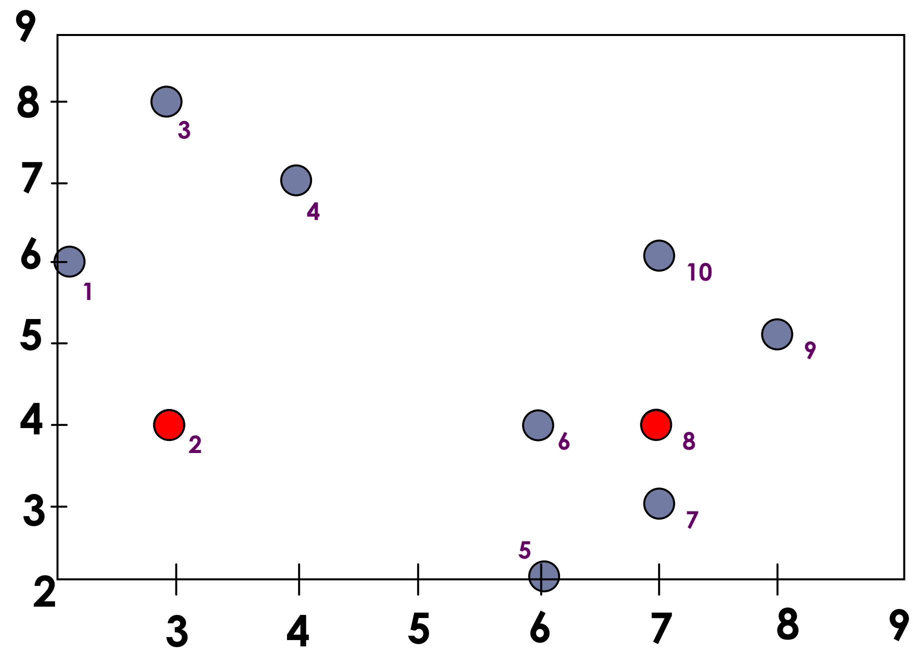
- Choose the configuration with the lowest cost

- Cost function is the difference in absolute error-value if a current representative object is replaced by a non-representative object

K-Medoids Method: Example

Data Objects

	A_1	A_2
O_1	2	6
O_2	3	4
O_3	3	8
O_4	4	7
O_5	6	2
O_6	6	4
O_7	7	3
O_8	7	4
O_9	8	5
O_{10}	7	6



Goal: create two clusters

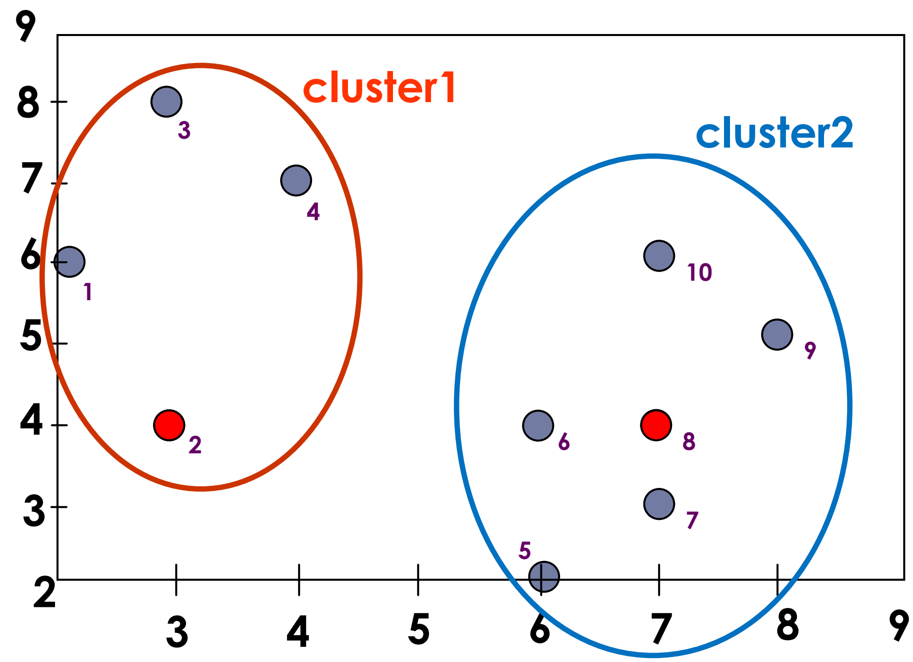
Choose randmly two medoids

$O_2 = (3,4)$
$O_8 = (7,4)$

K-Medoids Method: Example

Data Objects

	A_1	A_2
O_1	2	6
O_2	3	4
O_3	3	8
O_4	4	7
O_5	6	2
O_6	6	4
O_7	7	3
O_8	7	4
O_9	8	5
O_{10}	7	6



→Assign each object to the closest representative object

→Using L1 Metric (Manhattan), we form the following clusters

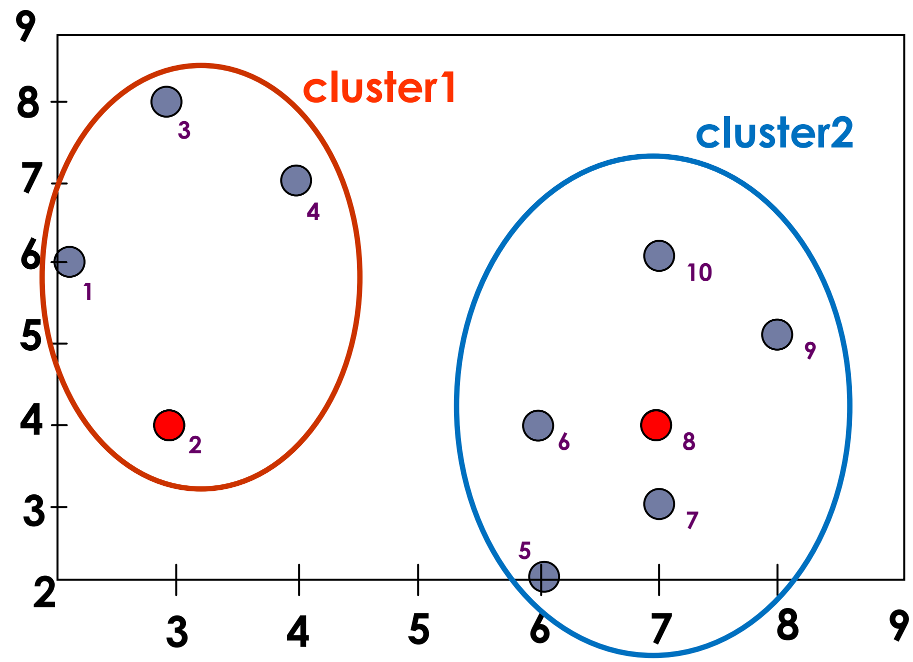
Cluster1 = $\{O_1, O_2, O_3, O_4\}$

Cluster2 = $\{O_5, O_6, O_7, O_8, O_9, O_{10}\}$

K-Medoids Method: Example

Data Objects

	A_1	A_2
O_1	2	6
O_2	3	4
O_3	3	8
O_4	4	7
O_5	6	2
O_6	6	4
O_7	7	3
O_8	7	4
O_9	8	5
O_{10}	7	6



→Compute the absolute error criterion **[for the set of Medoids (O2,O8)]**

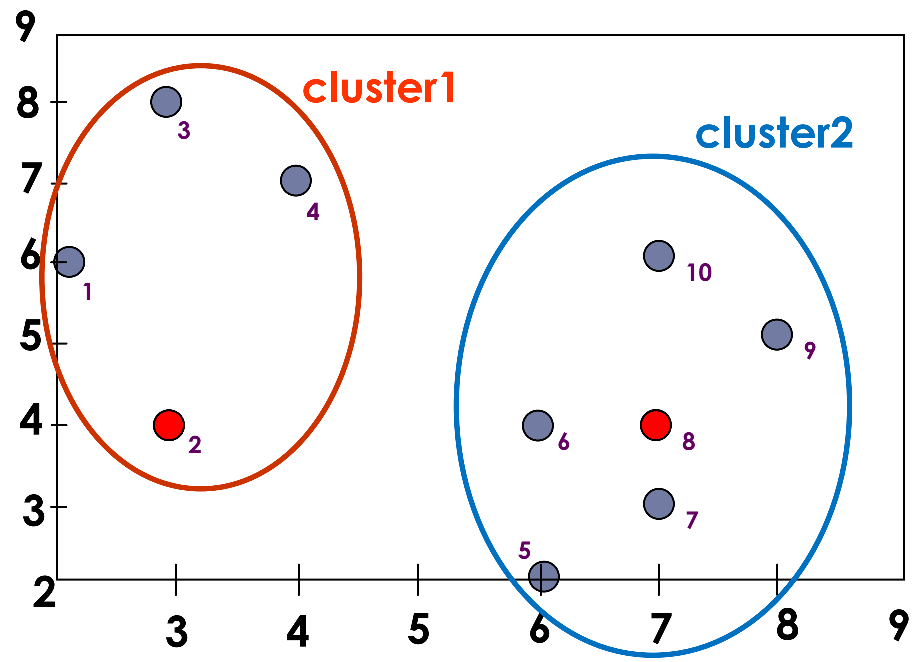
$$E=\sum_{i-1}^{k}\sum_{p\in C_i}|p-o_i|=|o_1-o_2|+|o_3-o_2|+|o_4-o_2|$$

$$+|o_5-o_8|+|o_6-o_8|+|o_7-o_8|+|o_9-o_8|+|o_{10}-o_8|$$

K-Medoids Method: Example

Data Objects

	A_1	A_2
O_1	2	6
O_2	3	4
O_3	3	8
O_4	4	7
O_5	6	2
O_6	6	4
O_7	7	3
O_8	7	4
O_9	8	5
O_{10}	7	6



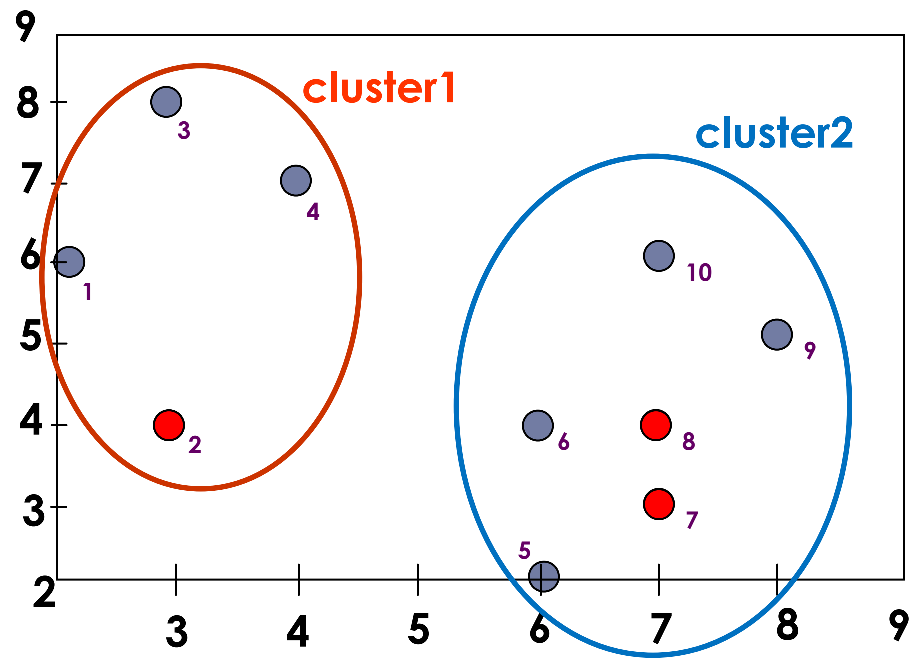
→The absolute error criterion **[for the set of Medoids (O2,O8)]**

$$E = (3+4+4) + (3+1+1+2+2) = 20$$

K-Medoids Method: Example

Data Objects

	A_1	A_2
O_1	2	6
O_2	3	4
O_3	3	8
O_4	4	7
O_5	6	2
O_6	6	4
O_7	7	3
O_8	7	4
O_9	8	5
O_{10}	7	6



→Choose a random object O_7

→Swap **O8** and **O7**

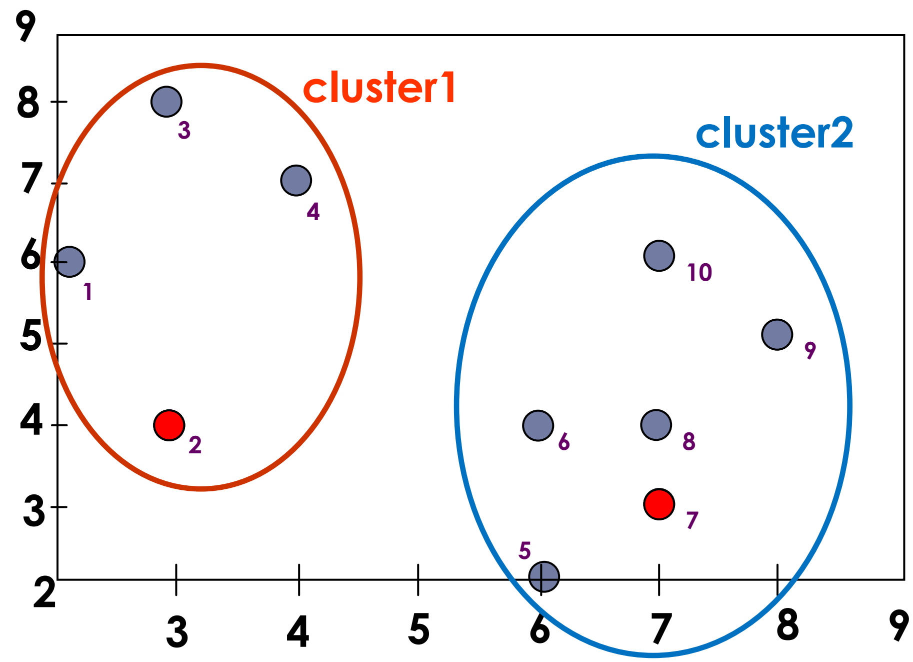
→Compute the absolute error criterion **[for the set of Medoids (O2,O7)]**

$$E = (3+4+4)+(2+2+1+3+3) = 22$$

K-Medoids Method: Example

Data Objects

	A_1	A_2
O_1	2	6
O_2	3	4
O_3	3	8
O_4	4	7
O_5	6	2
O_6	6	4
O_7	7	3
O_8	7	4
O_9	8	5
O_{10}	7	6



→Compute the cost function

Absolute error [for O_2,O_7] – Absolute error [O_2,O_8]

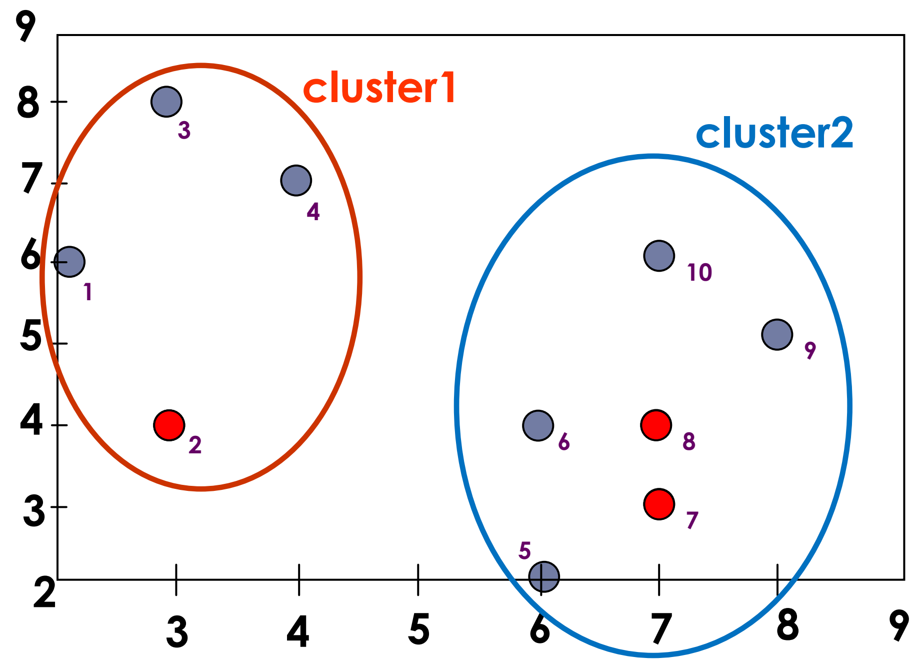
$$S = 22 - 20$$

S> 0 ⇒ it is a bad idea to replace O_8 by O_7

K-Medoids Method

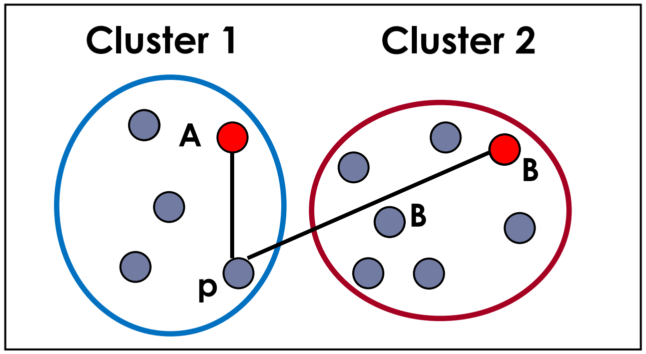
Data Objects

	A_1	A_2
O_1	2	6
O_2	3	4
O_3	3	8
O_4	4	7
O_5	6	2
O_6	6	4
O_7	7	3
O_8	7	4
O_9	8	5
O_{10}	7	6



- In this example, changing the medoid of cluster 2 did not change the assignments of objects to clusters.
- What are the possible cases when we replace a medoid by another object?

K-Medoids Method



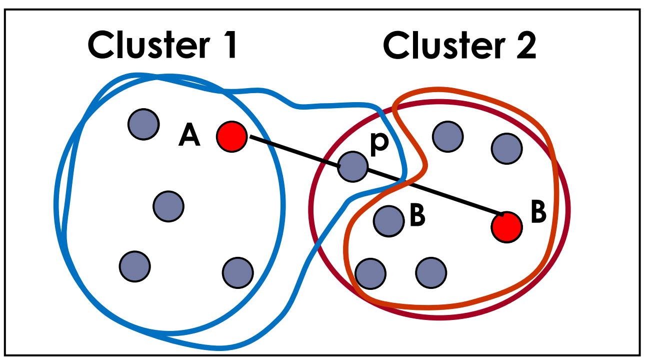
● Representative object

● Random Object

Currently **P** assigned to **A**

First case

The assignment of **P** to **A** does **not change**



● Representative object

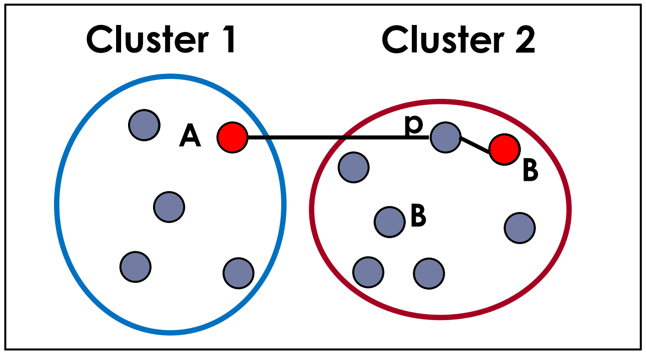
● Random Object

Currently **P** assigned to **B**

Second case

P is **reassigned** to **A**

K-Medoids Method



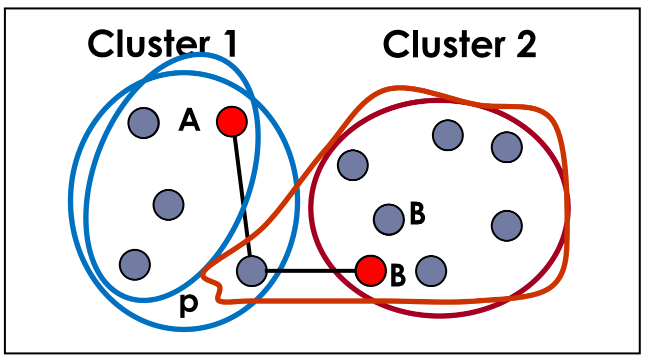
Representative object

Random Object

Currently **P** assigned to **B**

Third case

P is **reassigned** to **the new B**



Representative object

Random Object

Currently **P** assigned to **A**

Fourth case

P is **reassigned** to **B**

K-Medoids Algorithm(PAM)

PAM : Partitioning Around Medoids

- **Input**
  - K: the number of clusters
  - D: a data set containing n objects
- **Output**: A set of k clusters
- **Method**:

(1) Arbitrary choose k objects from D as representative objects (seeds)

(2) **Repeat**

(3) Assign each remaining object to the cluster with the nearest representative object

(4) For each representative object O_j

(5) Randomly select a non representative object O_{random}

(6) Compute the total cost **S** of swapping representative object Oj with O_{random}

(7) if $S<0$ then replace O_j with O_{random}

(8) **Until** no change

K-Medoids Properties(k-medoids vs.K-means)

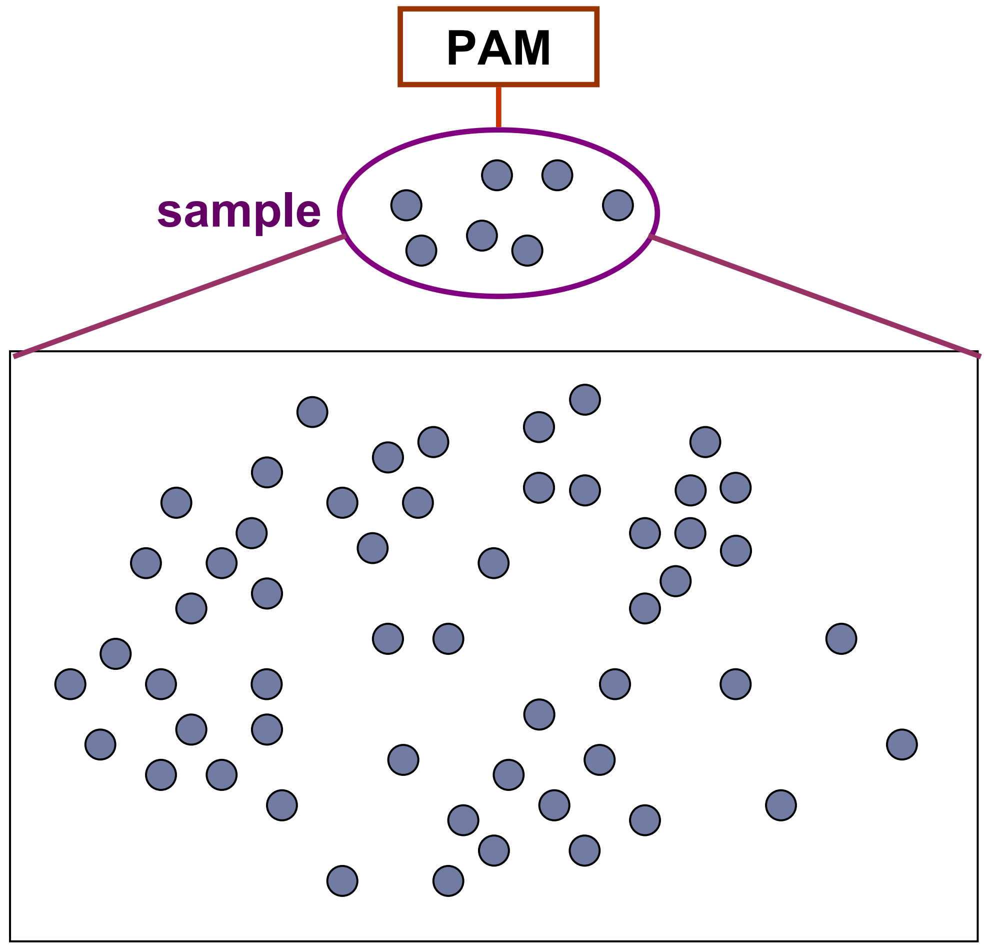
- The complexity of each iteration is **$O(k(n-k)^2)$**
- For large values of n and k, such computation becomes very costly
- **Advantages**
  - K-Medoids method is more robust than k-Means in the presence of noise and outliers
- **Disadvantages**
  - K-Medoids is more costly that the k-Means method
  - Like k-means, k-medoids requires the user to specify k
  - It does not scale well for large data sets

3.2.4 CLARA

- **CLARA (Clustering Large Applications)** uses a sampling-based method to deal with large data sets

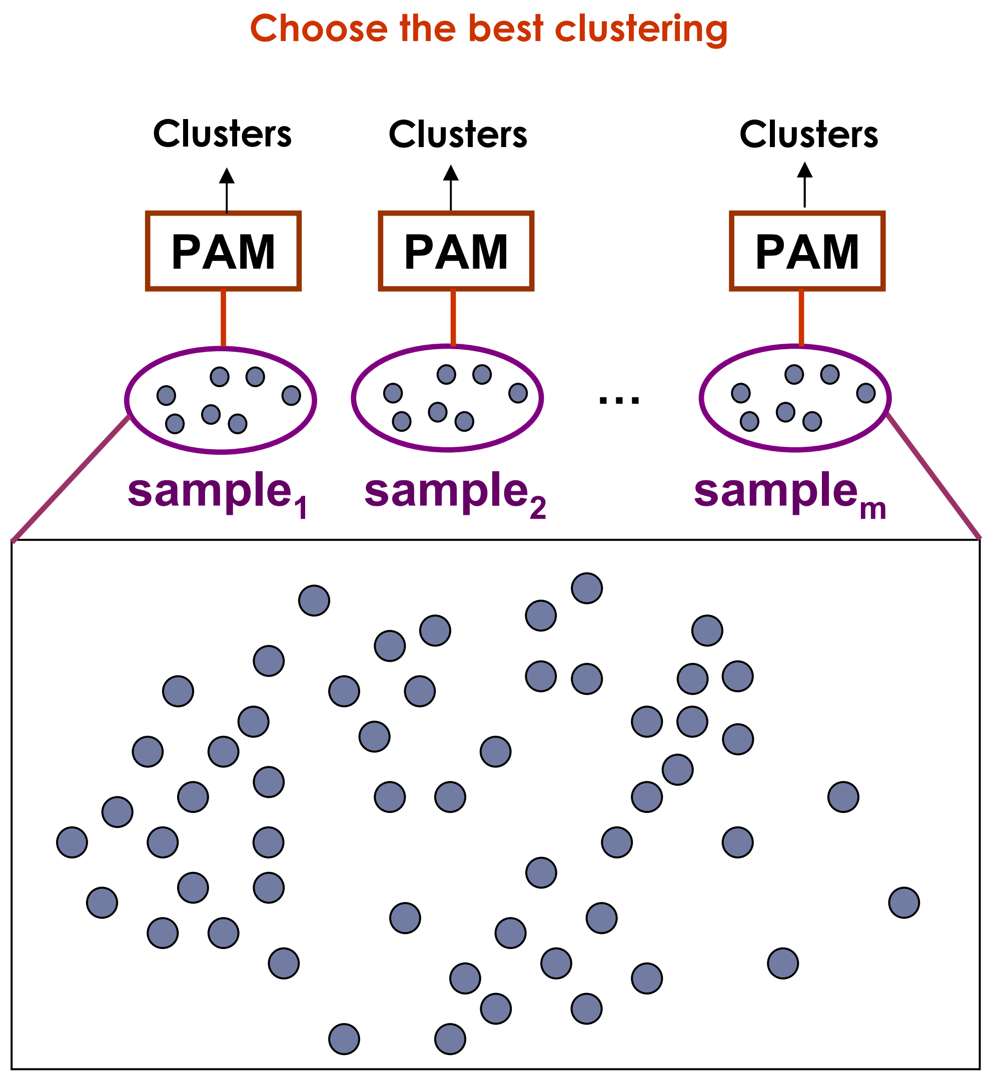
- A random sample should closely represent the original data

- The chosen medoids will likely be similar to what would have been chosen from the whole data set



CLARA

- Draw multiple samples of the data set
- Apply PAM to each sample
- Return the best clustering



CLARA Properties

- Complexity of each Iteration is: **$O(ks^2 + k(n-k))$**
  - **s**: the size of the sample
  - **k**: number of clusters
  - **n**: number of objects
- **PAM** finds the best k medoids among a given data, and **CLARA** finds the best k medoids among the selected samples
- **Problems**
  - The best k medoids may not be selected during the sampling process, in this case, CLARA will never find the best clustering
  - If the sampling is biased we cannot have a good clustering
  - Trade off-of efficiency

3.2.5 CLARANS

- **CLARANS (Clustering Large Applications based upon RANdomized Search)** was proposed to improve the quality and the scalability of CLARA

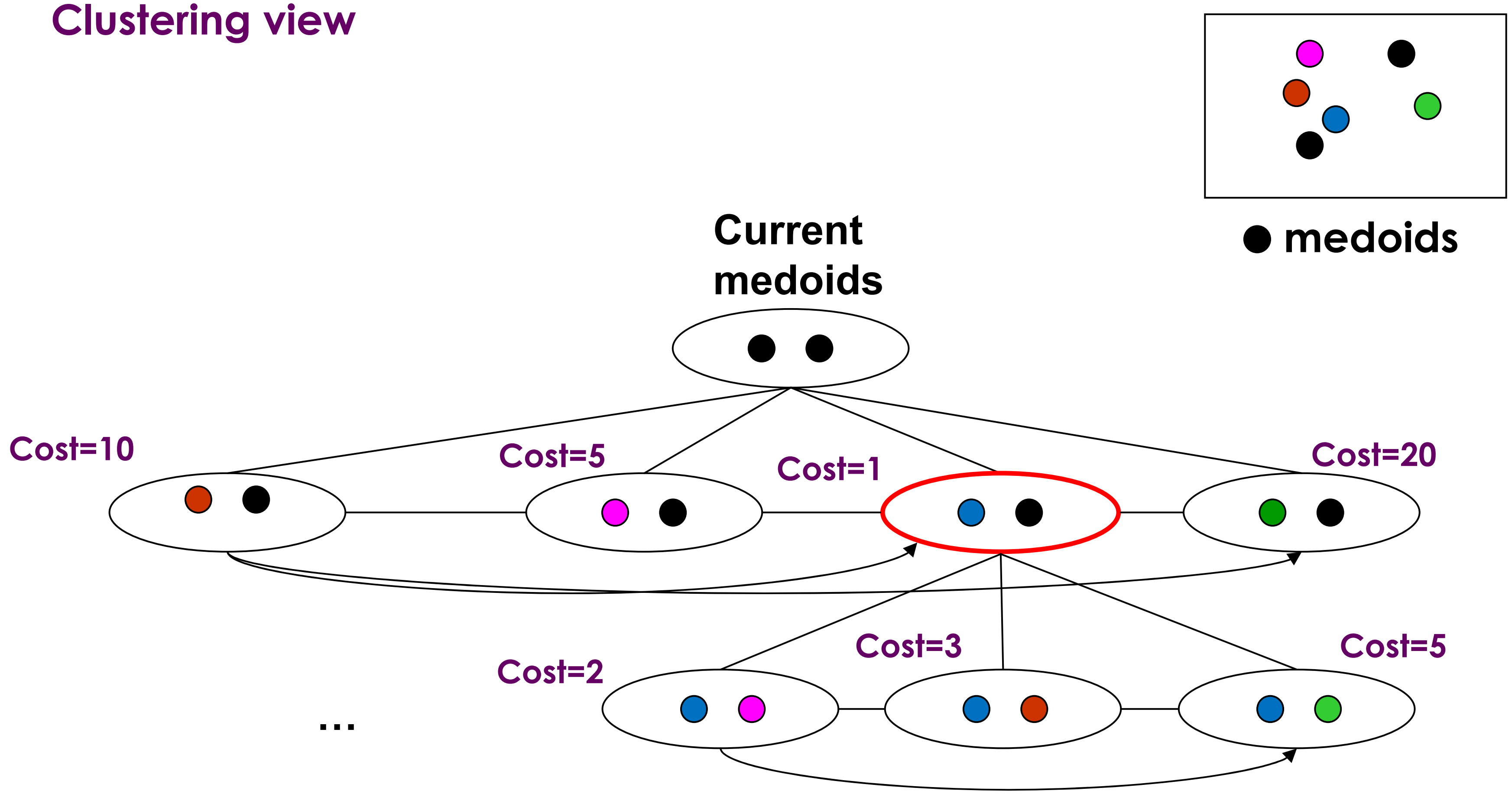
- It combines sampling techniques with PAM

- It does not confine itself to any sample at a given time

- It draws a sample with some randomness in each step of the search

CLARANS: The idea

Clustering view



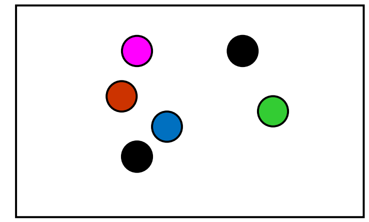
Keep the current medoids

CLARANS: The idea

CLARA

- Draws a sample of nodes at the beginning of the search
- Neighbors are from the chosen sample
- Restricts the search to a specific area of the original data

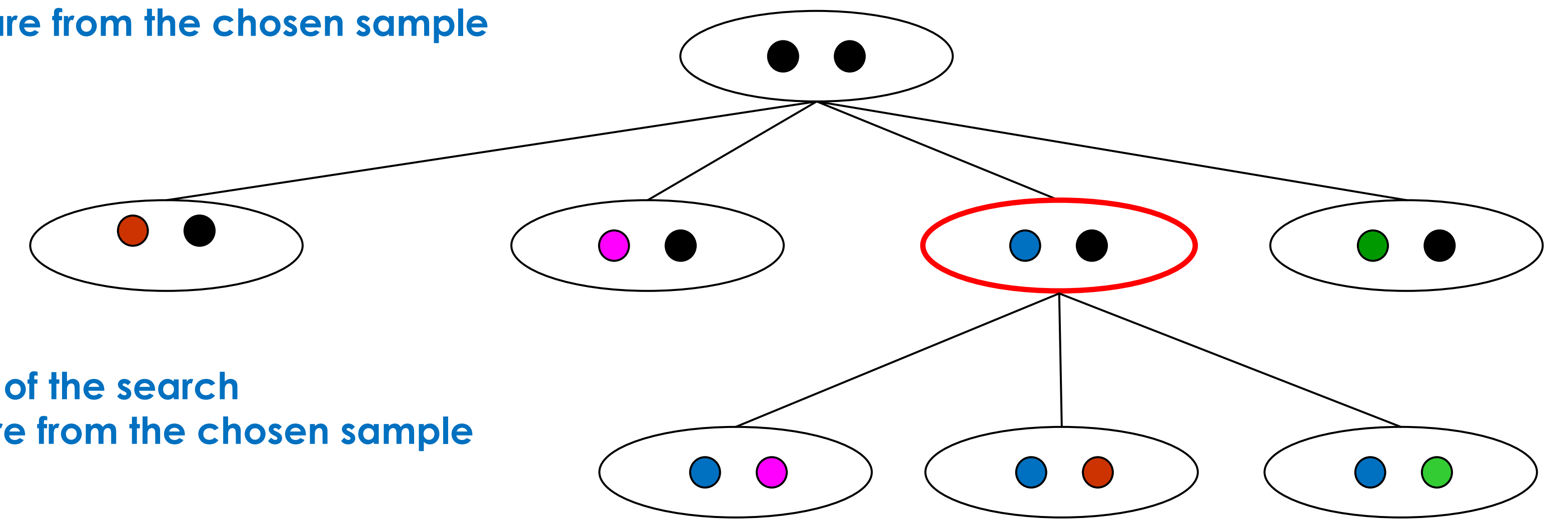
Sample



medoids

Current medoids

First step of the search
Neighbors are from the chosen sample



second step of the search
Neighbors are from the chosen sample

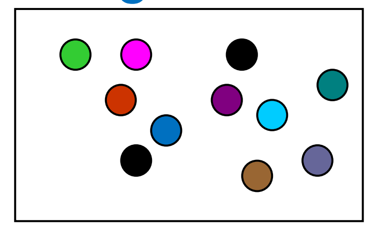
…

CLARANS: The idea

CLARANS

- Does not confine the search to a localized area
- Stops the search when a local minimum is found
- Finds several local optimums and output the clustering with the best local optimum

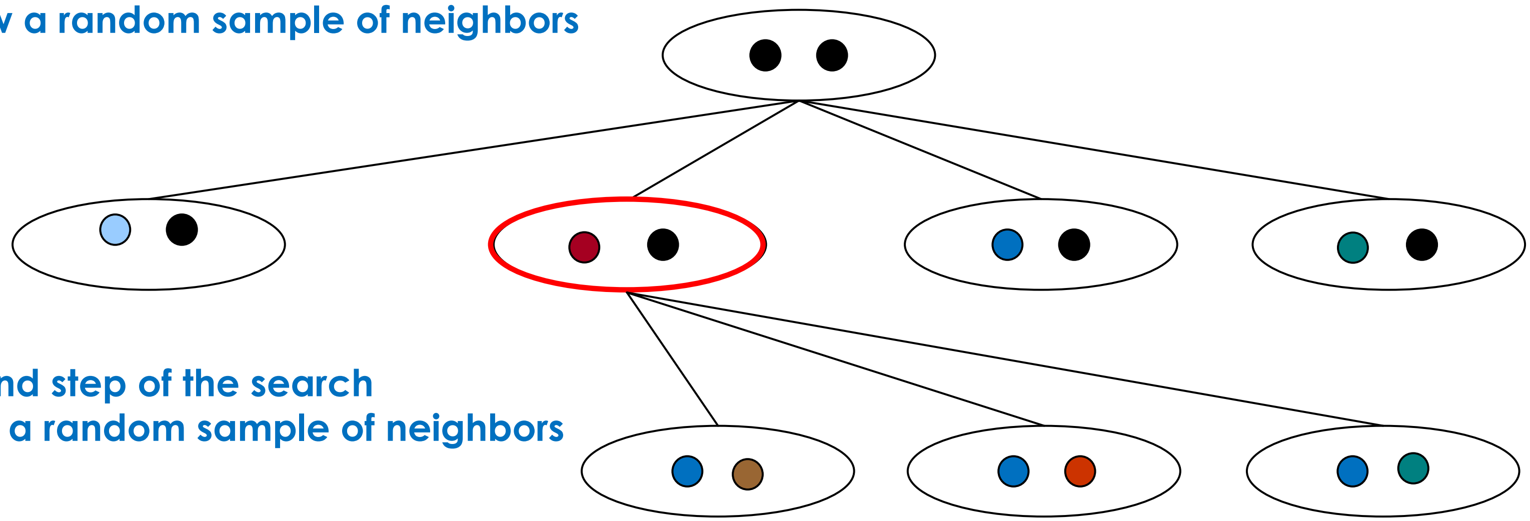
Original data



● **medoids**

First step of the search
Draw a random sample of neighbors

Current medoids



second step of the search
Draw a random sample of neighbors

…

The number of neighbors sampled from the original data is specified by the user

CLARANS Properties

- **Advantages**
  - → Experiments show that CLARANS is more effective than both PAM and CLARA
  - → Handles outliers
- **Disadvantages**
  - → The computational complexity of CLARANS is **$O(n^2)$**, where n is the number of objects
  - → The clustering quality depends on the sampling method

Summary of Section 3.2

- **Partitioning** methods find sphere-shaped clusters
- **K- mean** is efficient for large data sets but sensitive to outliers
- **PAM** uses centers of the clusters instead of means
- **CLARA** and **CLARANS** are used for clustering large databases